РЕКОМЕНДАТЕЛЬНЫЕ СИСТЕМЫ II

Сергей Николенко

TRA Robotics — Санкт-Петербург
28 июня 2018 г.

Random facts:

- 28 июня 1914 г. сначала Неделько Чабринович бросил гранату, но промахнулся, а вот браунинг Гаврилы Принципа через несколько часов был точен...
- ...а ровно через пять лет, 28 июня 1919 г., был подписан Версальский мирный договор и образована Лига Наций
- 28 июня 1956 г. в Познани на заводе имени Сталина начались демонстрации рабочих, требующих улучшения условий труда; массовые беспорядки «Познанского июня» были подавлены, но успели вдохновить молодых венгров
- 28 июня 1997 г. за 30 секунд до конца третьего раунда Эвандер Холифилд лишился дарвинова бугорка

Метрики качества и расширения

Метрики качества

- Ещё одна важная тема, которую в рекомендательных системах часто замалчивают.
- Как оценить качество рекомендаций? Какая должна быть метрика качества?
- Когда мы обучаем SVD (максимизируем правдоподобие), мы оптимизируем среднеквадратичную ошибку отклонения.
- И Netflix Prize, например, так и был сформулирован: надо было оптимизировать RMSE, среднеквадратичное отклонение предсказаний от истинного рейтинга пользователя.
- Но что нам надо на самом деле? Что у нас в тестовом множестве?

Метрики качества

- В тестовом множестве отложены рейтинги некоторых продуктов, которым пользователь дал оценку.
- Наша задача – выдать пользователю топ-рекомендации; не предсказать все рейтинги, а найти, у каких продуктов рейтинг будет самым большим.
- То есть на самом деле это задача *ранжирования*! И метрики качества лучше брать из information retrieval; там, когда оценивают качество выдачи, вовсе не пытаются минимизировать отклонение предсказания функции релевантности.
- Дальше для простоты будем рассматривать бинарный случай (лайк-дизлайк).

Метрики качества

- Классические метрики:
  (1) точность (precision) – количество «хороших» (релевантных запросу в случае поисковой выдачи, отмеченных высокой оценкой в случае рекомендательной системы) документов в выдаче, делённое на общее количество документов в выдаче;
  (2) полнота (recall) – количество «хороших» документов в выдаче, делённое на общее количество релевантных документов в базе поисковой системы.
- Однако здесь те же проблемы; эти параметры не зависят от ранжирования выдачи, надо знать заранее, сколько потребуется рекомендаций.

Метрики качества

- Метрики качества ранжирования:
  - NDCG, Normalized Discounted Commulative Gain; выберем топ-k рекомендаций (k может быть заведомо больше нужного числа) и посчитаем:

$$\mathrm{DCG}_k = \sum_{i=1}^{k} \frac{2^{\hat{r}_i} - 1}{\log_2(1+i)},$$

$$\mathrm{NDCG}_k = \frac{\mathrm{DCG}_k}{\mathrm{IDCG}_k},$$

    где $\hat{r}_i$ – наша оценка рейтинга продукта на позиции i, а IDCG_k – значение DCG_k при ранжировании по истинным значениям (рейтингам из валидационного набора);
  - NDCG от 0 до 1, но ей трудно придумать естественную интерпретацию (как вероятность чего-нибудь, например).

Метрики качества

- Метрики качества ранжирования:
  - AUC, Area Under (ROC) Curve; можно считать по всей выдаче сразу;
  - AUC – вероятность того, что случайно выбранная пара продуктов с разными оценками будет отранжирована правильно (понравившийся будет выше в выдаче, чем не понравившийся);
  - в бинарном случае можно посчитать в замкнутом виде:

$$\hat{A} = \frac{S_0 - n_0(n_0+1)/2}{n_0 n_1},$$

где n_0, n_1 – число понравившихся и не понравившихся пользователю объектов, $S_0 = \sum p_i$ – сумма номеров позиций понравившихся объектов в выдаче.

Метрики качества

- Метрики качества ранжирования:
  - но на самом деле простые метрики тоже важны, потому что обычно пользователь успевает увидеть только несколько самых верхних рекомендаций;
  - WTA (winner takes all) – эта метрика равна 1, если топ-рекомендация (с самым большим предсказанным рейтингом) из просмотренных пользователем получила положительную оценку, и 0 в противном случае;
  - Top$k$ – доля положительных оценок среди топ-k рекомендаций (Top10 часто называют MAP – mean average precision).

Регрессия по признакам

- Проблема: холодный старт.
- Надо как-то инициализировать; если вообще ничего не знаем, сделать ничего нельзя, конечно.
- Но так не бывает; обычно есть набор признаков – можно пытаться предсказывать значения факторов:
  - просто регрессией по признакам;
  - (обычно для продуктов) выделяя темы при помощи topic modeling.

Регрессия по признакам

- Получается, что для признаков пользователя x_i и продукта x_a мы рассматриваем модель

$$r_{i,a} \sim \mu + b_{\text{user}}(x_i) + b_{\text{item}}(x_a) + q_a^\top p_i(t),$$

где

$$b_{\text{user}}(x_i) \sim N(u(x_i), \sigma_u^2),$$
$$b_{\text{item}}(x_i) \sim N(v(x_i), \sigma_v^2),$$

и в качестве u и v может выступать любая регрессия [Agarwal, Chen, 2009].

Регрессия по признакам

- Ещё один вариант, через контент:
  - выделить темы из продуктов (LDA), получится распределение $z_{a,k}$ для каждого a;
  - обучить факторы $s_{i,k}$ того, насколько пользователю “нравятся” эти темы;
  - затем для нового продукта оценить темы $\hat{z}_{a,k}$ по контенту, а потом добавлять в модель слагаемое

$$r_{i,a} \sim \ldots + \sum_k s_{i,k}\hat{z}_{a,k},$$

    что помогает для холодного старта по продуктам.
- Есть модели, связанные с тем, как обучить не абы какие темы, а хорошо выражающие предпочтения (fLDA).

Время в коллаборативной фильтрации

- Пример: давайте добавим время, т.е. будем рассматривать базовые предикторы и характеристики пользователя как функции от времени:

$$\hat{r}_{i,a} = \mu + b_i(t) + b_a(t) + q_a^\top p_i(t),$$

где

$$\begin{aligned}
b_a(t) =& b_a + b_{a,\mathrm{Bin}(t)}, \\
b_i(t) =& b_i + \alpha_i \mathrm{dev}_i(t) + b_{i,t}, \\
p_{i,f}(t) =& p_{i,f} + \alpha_{i,f}\mathrm{dev}_i(t) + p_{i,f,t} + \frac{1}{\sqrt{|V(i)|}} \sum_{b \in V(i)} y_b, \\
\mathrm{dev}_i(t) =& \mathrm{sign}(t - t_i)\,|t - t_i|^\beta .
\end{aligned}$$

- Это называется timeSVD++, и эта модель была одним из основных компонентов модели, взявшей Netflix Prize.

Социальные сети

- Предположим, что пользователи приходят из социальной сети.
- Т.е. есть друзья, есть социальный граф (его часть) и т.д. Это тоже можно добавить в рекомендательную модель:
  - фильтр/перевзвешивание в методе ближайших соседей;
  - дополнительные слагаемые в разложение типа SVD;
  - разложение матрицы доверия (из социального графа) вместе с матрицей рейтингов, меняем априорное распределение для PMF и т.д.

Метрики разнообразия

- Filter bubble: как вывести человека за его привычный круг.
- Можно до конца жизни рекомендовать одно и то же; метрики:
  - diversity – разнообразие, мера похожести элементов списка;
  - novelty – новизна для пользователя, распространённость продукта, доля его рейтингов;
  - serendipity – неожиданность, сюрприз, похожесть на историю пользователя.
- Для всего этого нужно уметь распознавать похожесть контента рекомендованных товаров.

Контекстно-зависимые рекомендации

- CARS (context-aware recommender systems) – мы рекомендуем в контексте:
  - временном;
  - ситуативном;
  - географическом;
  - предшествующего поведения пользователей и т.д.

Контекстно-зависимые рекомендации

- Формально контекст – это новые измерения в матрице предпочтений.
- Получается “гиперкуб” данных, есть методы тензорного разложения, аналогичного SVD.
- Но часто не хуже работают простые решения – отфильтровать контексты и обучить модели только по этим данным, добавить полученные модели и сам контекст как факторы в бленд.

Онлайн-модели

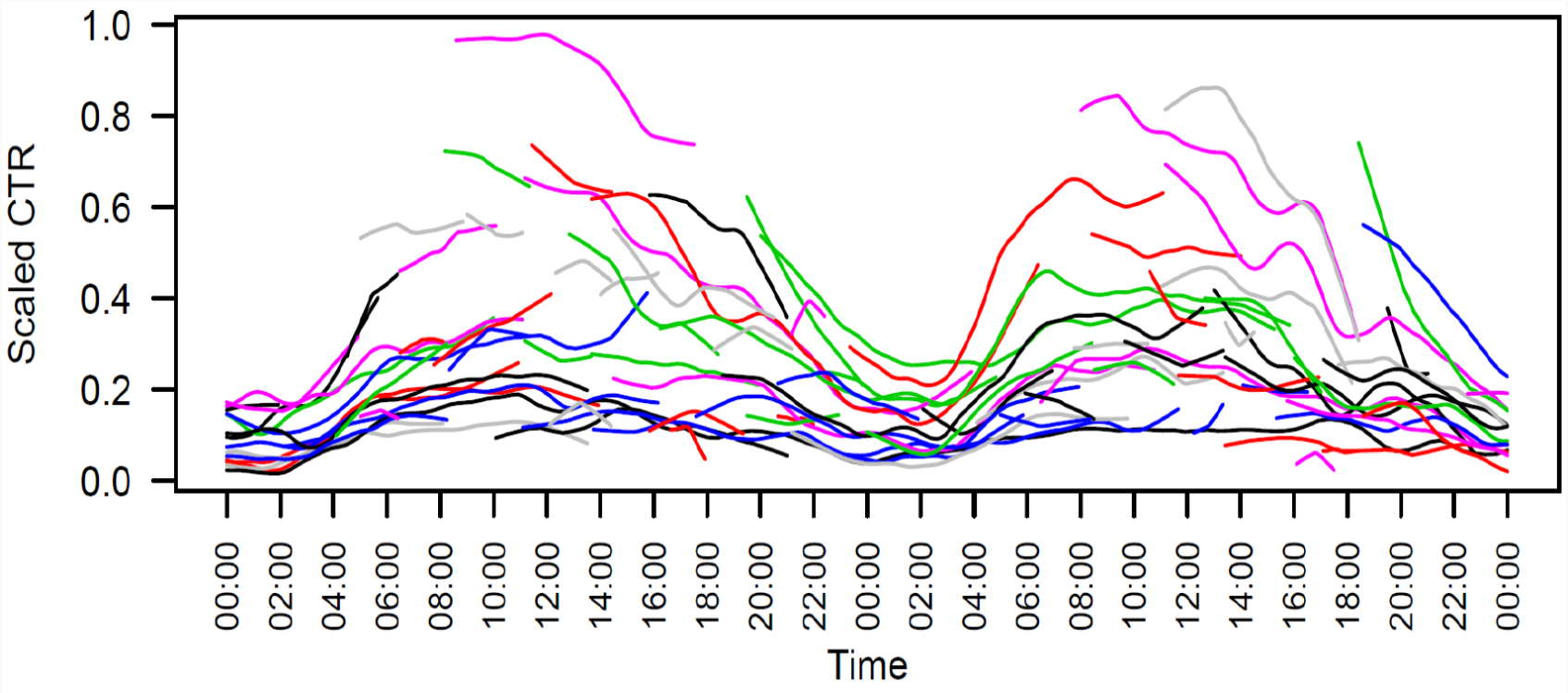
Постановка задачи

- *Онлайн-модели* отличаются от оффлайн-моделей тем, что их главная цель – как можно быстрее «поймать» изменения популярности тех или иных продуктов.
- Данных тут недостаточно, чтобы такие изменения можно было поймать методами коллаборативной фильтрации.
- Поэтому онлайн-методы обычно меньше персонализированы, индивидуальных данных не набёрется просто.

Постановка задачи

- Казалось бы, что может быть проще – есть набор продуктов/сайтов $a_1, \dots, a_M$ со средними рейтингами $\bar{r}_1, \dots, \bar{r}_M$; давайте упорядочим их по среднему рейтингу и будем рекомендовать пользователю продукты с наивысшим средним рейтингом.
- Однако такая система не будет достаточно чувствительной к быстрым изменениям истинного среднего рейтинга: представьте, что $\bar{r}_i$ внезапно резко уменьшился – может понадобиться очень много новых показов, чтобы привести нашу его оценку в соответствие с новым значением.

Пример


Scaled CTR
1.0
0.8
0.6
0.4
0.2
0.0
00:00
02:00
04:00
06:00
08:00
10:00
12:00
14:00
16:00
18:00
20:00
22:00
00:00
02:00
04:00
06:00
08:00
10:00
12:00
14:00
16:00
18:00
20:00
22:00
00:00
Time

Рекомендатели и бандиты

- Если мы хотим быстро оценивать средний рейтинг, то мы попадём в ситуацию обучения с подкреплением: есть набор продуктов, их надо рекомендовать, исход заранее неизвестен, и мы хотим оптимизировать суммарный рейтинг.
- Это называется задачей о *многоруких бандитах* (multiarmed bandits).
- О них мы скоро поговорим куда подробнее, а пока рассмотрим несколько другую модель...

DGP

- Модель Dynamic Gamma–Poisson (DGP): фиксируем период времени t (небольшой) и будем считать показы и клики (рейтинги, отметки «like» и т.д.) за время t.
- Пусть мы в течение периода t показали продукт n_t раз и получили суммарный рейтинг r_t (если это ссылки на странице, например, то будет суммарное число кликов $r_t \leq n_t$).
- Тогда нам в каждый момент t дана последовательность $n_1, r_1, n_2, r_2, \ldots, n_t, r_t$, и мы хотим предсказать p_{t+1} (доля успешных показов в момент $t+1$, CTR).

DGP

- Вероятностные предположения модели DGP:
  (1) $(r_t \mid n_t, p_t) \sim \mathrm{Poisson}(n_t, p_t)$
  (для данных n_t и p_t r_t имеет пуассоновское распределение);
  (2) $p_t = \epsilon_t p_{t-1}$, где $\epsilon_t \sim \mathrm{Gamma}(\mu = 1, \sigma = \eta)$
  (средняя доля успешных показов p_t меняется не слишком быстро, а путём умножения на случайную величину ϵ_t, которая имеет гамма-распределение вокруг единицы);
  (3) параметрами модели являются параметры распределения $p_1 \sim \mathrm{Gamma}(\mu = \mu_0, \sigma = \sigma_0)$, а также параметр η, который показывает, насколько «гладко» может изменяться p_t;
  (4) задача — оценить параметры апостериорного распределения

$$(p_{t+1} \mid n_1, r_1, n_2, r_2, \dots, n_t, r_t) \sim \mathrm{Gamma}(\mu = ?, \sigma = ?).$$

DGP

- Можно пересчёт параметров в этой модели явно вычислить аналитически.
- Пусть на предыдущем шаге $t-1$ мы получили некоторую оценку μ_t, σ_t для параметров модели:

$$(p_t \mid n_1, r_1, n_2, r_2, \dots, n_{t-1}, r_{t-1}) \sim \mathrm{Gamma}(\mu = \mu_t, \sigma = \sigma_t),$$

  а затем получили новую точку (n_t, r_t).
- Тогда, обозначив $\gamma_t = \frac{\mu_t}{\sigma_t^2}$ (эффективный размер выборки), сначала уточним оценки μ_t, σ_t:

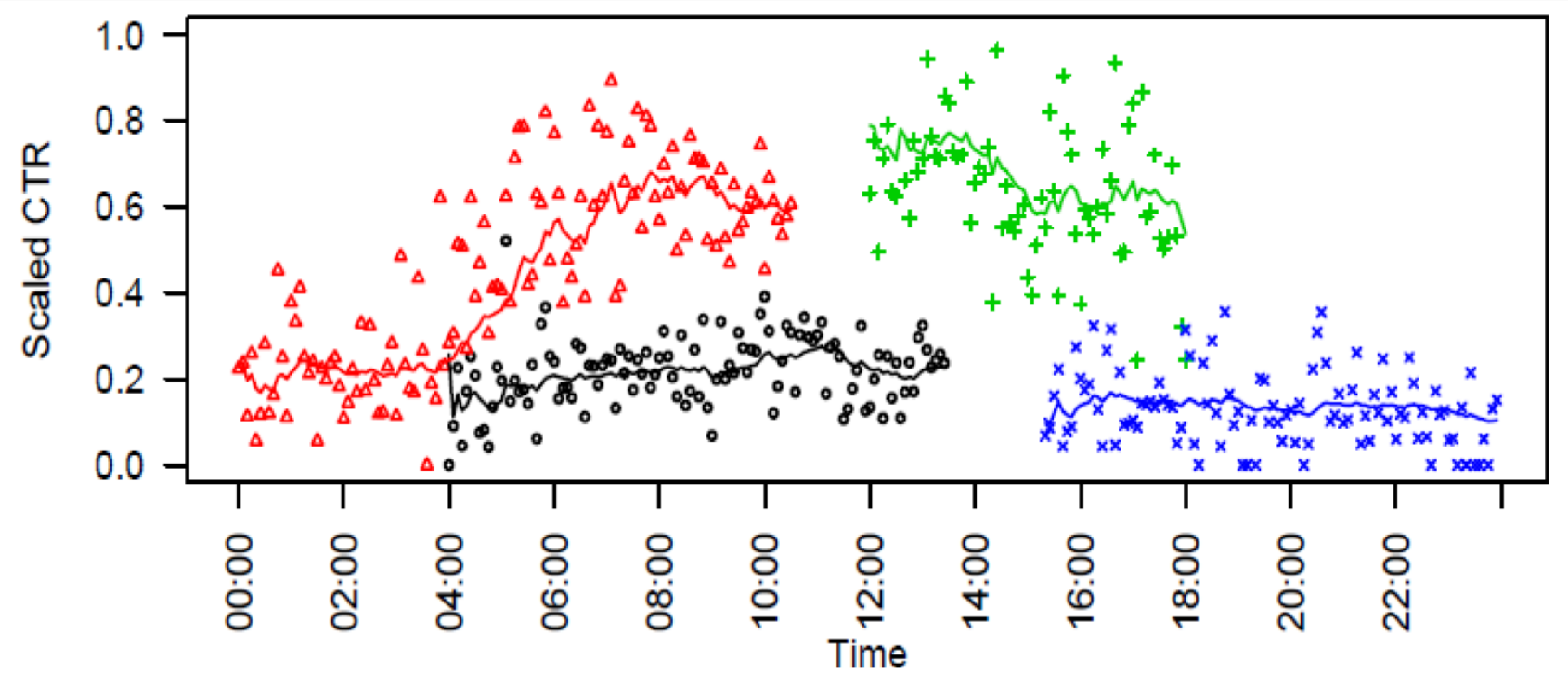
$$\gamma_{t|t} = \gamma_t + n_t, \quad \mu_{t|t} = \frac{\mu_t \gamma_t + r_t}{\gamma_{t|t}}, \quad \sigma^2_{t|t} = \frac{\mu_{t|t}}{\gamma_{t|t}}.$$

DGP

- А затем породим новое предсказание для $(p_{t+1} \mid n_1, r_1, \dots, n_t, r_t)$:

$$\begin{aligned}\mu_{t+1} &= \mu_{t|t}, \\ \sigma^2_{t+1} &= \sigma^2_{t|t} + \eta\left(\mu^2_{t|t} + \sigma^2_{t|t}\right).\end{aligned}$$

Пример


Scaled CTR
1.0
0.8
0.6
0.4
0.2
0.0
00:00
02:00
04:00
06:00
08:00
10:00
12:00
14:00
16:00
18:00
20:00
22:00
Time

Априорное распределение

- Тут интересный вопрос – чем инициализировать; поскольку надо всё делать быстро, хорошее априорное распределение очень важно.
- Пусть в тестовой выборке N записей, для которых известны показатели $r_1^{(i)}$ и $n_1^{(i)}$ (число показов и успешных показов за первый период времени); мы хотим получить оценку μ_0 и σ_0 для нового, неизвестного сайта, которая должна хорошо аппроксимировать ожидаемое r_1 и n_1 для нового сайта.
- Тогда ответ такой – её нужно считать как

$$\arg\max_{\mu_0,\sigma_0}\left[N\frac{\mu_0^2}{\sigma_0^2}\log\frac{\mu_0}{\sigma_0^2} - N\log\mathrm{Gamma}\left(\frac{\mu_0^2}{\sigma_0^2}\right) + \right.$$
$$\left. + \sum_i\left(\log\mathrm{Gamma}\left[r_1^{(i)} + \frac{\mu_0^2}{\sigma_0^2}\right] - \left[r_1^{(i)} + \frac{\mu_0^2}{\sigma_0^2}\right]\log\left[n_1^{(i)} + \frac{\mu_0}{\sigma_0^2}\right]\right)\right].$$

Что дальше

- Здесь тоже масса активно развивающихся направлений.
  1. Как совместно оптимизировать сразу много показываемых элементов? Представьте себе homepage большого портала.
  2. Какова на самом деле целевая метрика? CTR – это средство, а не цель. Для портала – user retention? проведённое на портале время? доход рекламодателей?
  3. Как именно персонализировать – кластеризовать пользователей? как именно сглаживать? каковы признаки пользователей?

Краткое резюме

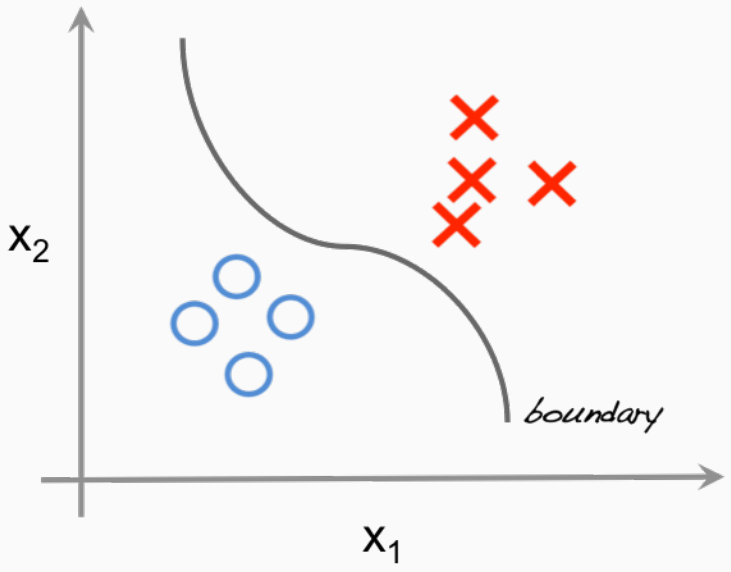
1. Метод ближайших соседей: GroupLens. User-based и item-based.
2. SVD-разложение матриц. Градиентный спуск и ALS. Байесовская версия (PMF).
3. Машины Больцмана – модель пользователя. Другие разложения: NMF.
4. Метрики качества: NDCG, AUC, Top-N.
5. Холодный старт и дополнительная информация: время, контекст etc.
6. Онлайн-системы (поиск трендов): DGP.

Обучение с подкреплением

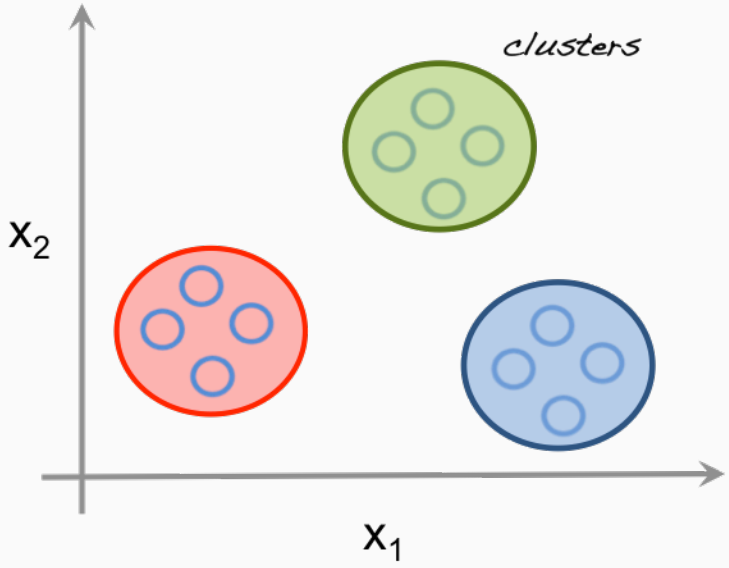
Постановка задачи

- В машинном обучении задача обычно ставится так:
  - или есть набор «правильных ответов», и нужно его продолжить на всё пространство (supervised learning),
  - или есть набор тестовых примеров без дополнительной информации, и нужно понять его структуру (unsupervised learning).

Supervised learning



Unsupervised learning

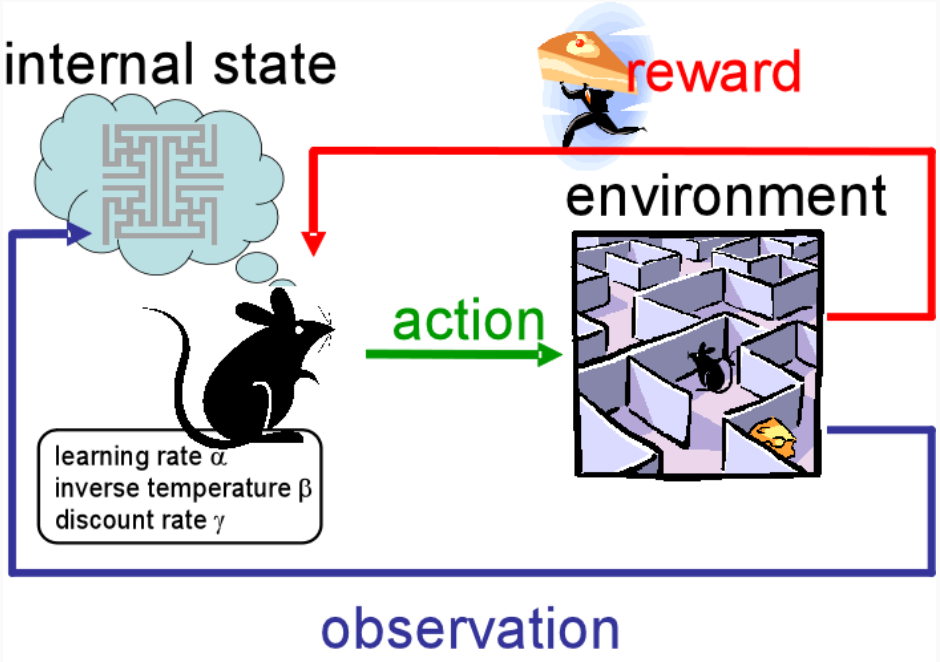


Постановка задачи

- Но как работает обучение в реальной жизни?
- Как ребёнок учится ходить?
- Мы далеко не всегда знаем набор правильных ответов, мы просто делаем то или иное действие и получаем результат.

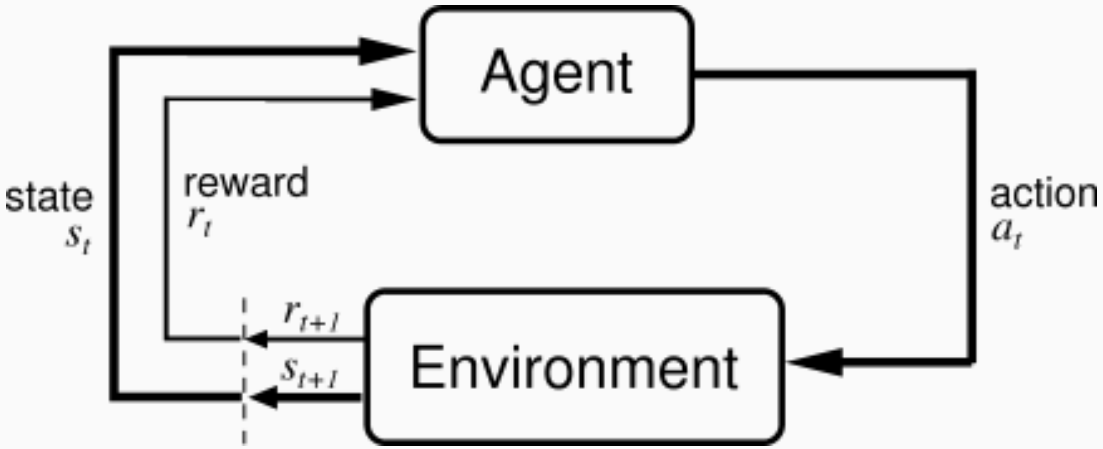
Постановка задачи

- Отсюда и обучение с подкреплением (reinforcement learning).
- Агент взаимодействует с окружающей средой, предпринимая действия; окружающая среда его поощряет за эти действия, а агент продолжает их предпринимать.



Постановка задачи -- формально

- На каждом шаге агент может находиться в состоянии $s \in S$.
- На каждом шаге агент выбирает из имеющегося набора действий некоторое действие $a \in A$.
- Окружающая среда сообщает агенту, какую награду r он за это получил и в каком состоянии s' после этого оказался.



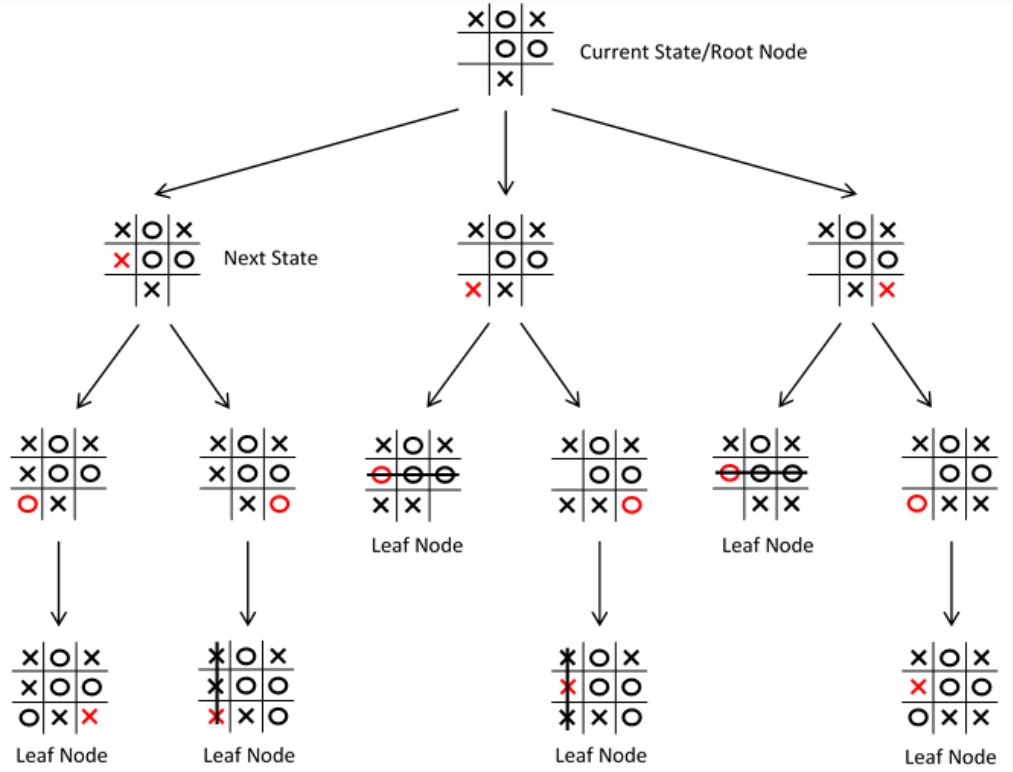
- (Sutton, Barto, 1998)

Exploitation vs. exploration

- Каждый алгоритм должен и изучать окружающую среду, и пользоваться своими знаниями, чтобы максимизировать прибыль.
- Вопрос — как достичь оптимального соотношения? Та или иная стратегия может быть хороша, но вдруг она не оптимальная?
- Этот вопрос всегда присутствует в обучении с подкреплением.

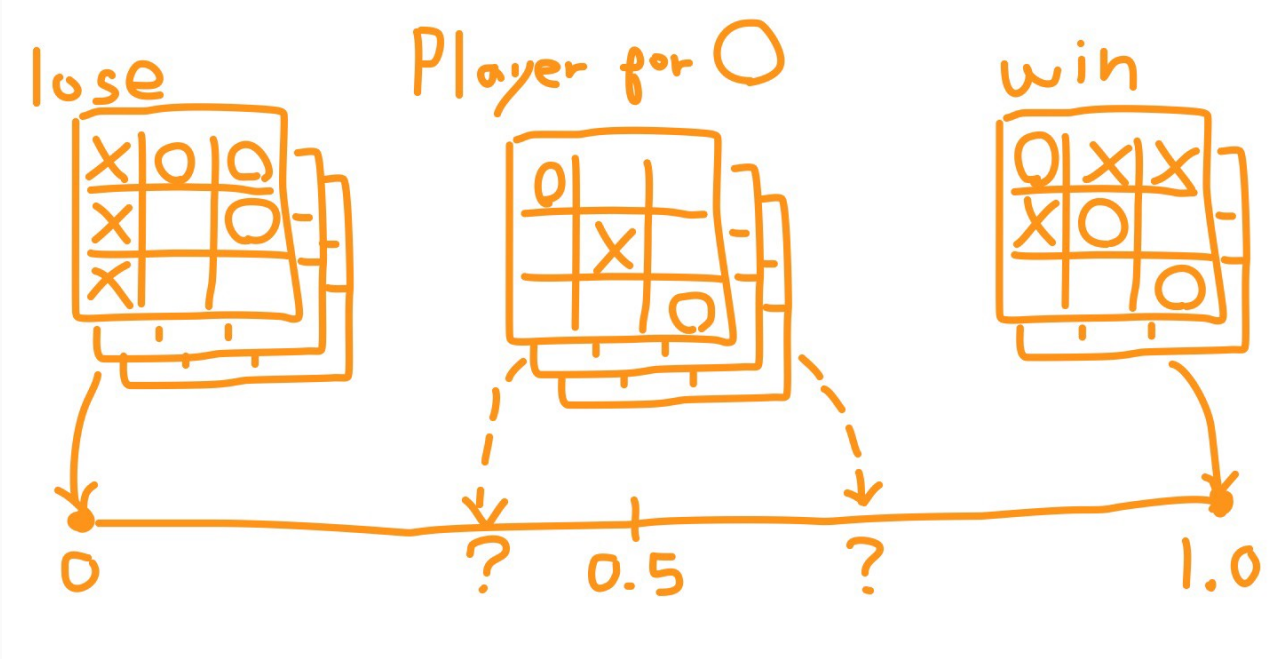
Пример

- Пример: крестики-нолики. Как научить машину играть и *выигрывать* в крестики-нолики?
- Можно, конечно, дерево построить, но это не масштабируется.



Пример

- Состояния – позиции на доске.
- Для каждого состояния введём функцию $V(s)$ (value function).
- Подкрепление приходит только в самом конце, когда мы выиграли или проиграли; как его распространить на промежуточные позиции?



Пример

- Небольшой трейлер того, что будет дальше: можно пропагировать оценку позиции обратно.
- Если мы попадали из s в s', апдейтим

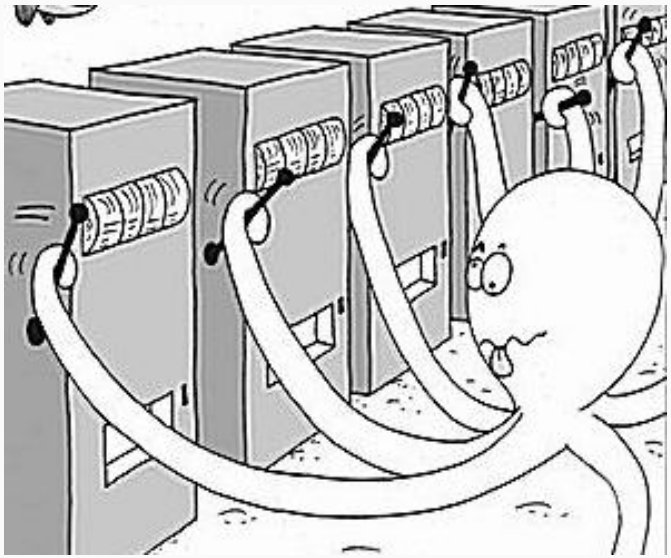
$$V(s) := V(s) + \alpha \left[V(s') - V(s) \right].$$

- Это называется TD-обучение (temporal difference learning), оно очень хорошо работает на практике и лежит в основе и AlphaGo, и много чего ещё.
- Но это будет только в конце нашего сегодняшнего пути...

Многорукие бандиты

Агенты с одним состоянием

- Формально всё то же самое, но $|S| = 1$, т.е. состояние агента не меняется. У него фиксированный набор действий A и возможность выбора из этого набора действий.
- Модель: агент в комнате с несколькими игровыми автоматами. У каждого автомата своё ожидание выигрыша.
- Нужно заработать побольше: exploration vs. exploitation.



Жадный алгоритм

- *Жадный алгоритм*: всегда выбирать стратегию, максимизирующую прибыль; прибыль можно оценить как среднее вознаграждение, полученное от этого действия:

$$Q_t(a) = \frac{r_1 + r_2 + ... + r_{k_a}}{k_a}.$$

- Что не так с таким алгоритмом?

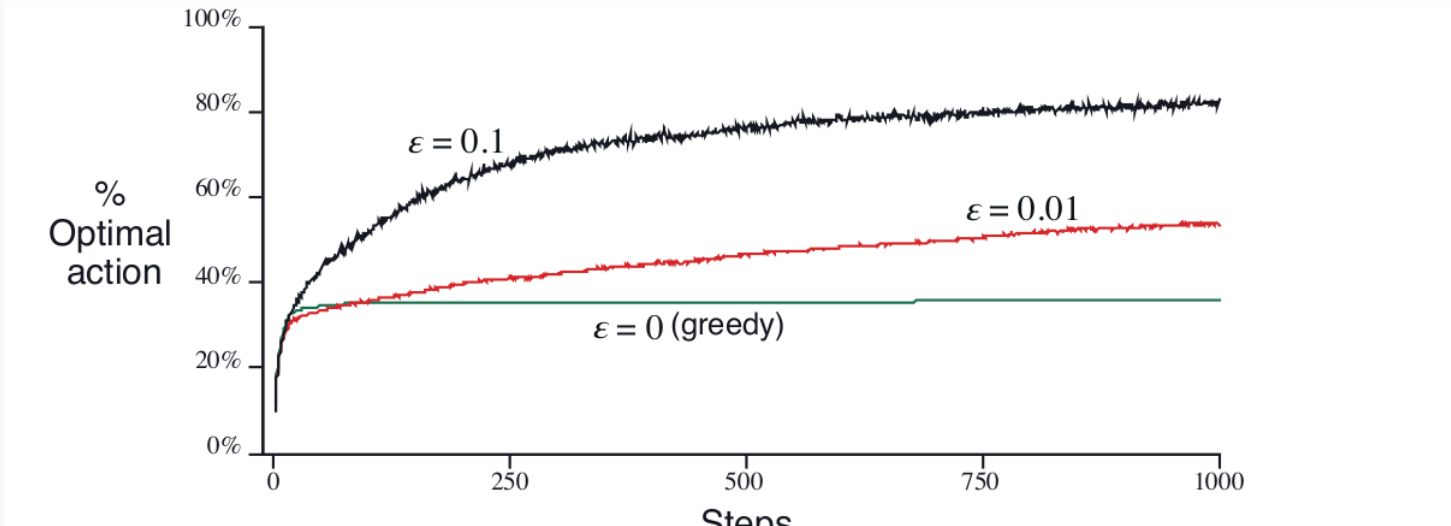
Жадный алгоритм

- Оптимум легко проглядеть, если на начальной выборке не повезёт (что вполне возможно).
- Поэтому полезная эвристика — *оптимизм при неопределённости*.

- То есть выбирать жадно, но при этом прибыль оценивать оптимистично, и нужны серьёзные свидетельства, чтобы отклонить стратегию.

Случайные стратегии

- *ϵ-жадная стратегия* (ϵ-greedy): выбрать действие с наилучшей ожидаемой прибылью с вероятностью $1 - \epsilon$, а с вероятностью ϵ выбрать случайное действие.



Интервальные оценки

- Естественный способ применить оптимистично–жадный метод – доверительные интервалы.
- Для каждого действия мы храним статистику n и w, а потом вычисляем доверительный интервал для вероятности успеха (с границей $1-\alpha$) и для выбора стратегии используем *верхнюю границу* этого интервала.
- Например, для испытаний Бернулли (монетка) с вероятностью $.95$ среднее лежит в интервале

$$\left(\bar{x} - 1.96\frac{s}{\sqrt{n}}, \bar{x} + 1.96\frac{s}{\sqrt{n}}\right),$$

  где 1.96 берётся из распределения Стьюдента, n — количество испытаний, $s = \sqrt{\frac{\sum(x-\bar{x})^2}{n-1}}$.
- Отличный метод, если вероятностные предположения соответствуют действительности (часто непонятно).

Правило инкрементального обновления

- Теперь – о среднем и $Q_t(a)$.
- Как пересчитывать $Q_t(a) = \frac{r_1 + \ldots + r_{k_a}}{k_a}$ при поступлении новой информации?
- Довольно просто:

$$Q_{k+1} = \frac{1}{k+1} \sum_{i=1}^{k+1} r_i = \frac{1}{k+1} \left[r_{k+1} + \sum_{i=1}^{k} r_i \right] =$$
$$= \frac{1}{k+1} \left(r_{k+1} + kQ_k \right) = Q_k + \frac{1}{k+1} \left(r_{k+1} - Q_k \right).$$

Правило инкрементального обновления

- Это частный случай общего правила – сдвигаем оценку так, чтобы уменьшалась ошибка:

  $$\text{НоваяОценка} := \text{СтараяОценка} + \text{Шаг}\,[\text{Цель} - \text{СтараяОценка}]\,.$$

- Заметим, что шаг у среднего непостоянный, $\alpha_k(a) = \frac{1}{k_a}$:

  $$Q_{k+1} = Q_k + \frac{1}{k+1}\left(r_{k+1} - Q_k\right).$$

- Изменяя последовательность шагов, можно добиться других эффектов.

Нестационарная задача

- Часто бывает, что выплаты из разных бандитов на самом деле нестационарны, т.е. меняются со временем.
- В такой ситуации имеет смысл давать большие веса недавней информации и маленькие веса – давней.
- Пример: у правила апдейта

$$Q_{k+1} = Q_k + \alpha \left[r_{k+1} - Q_k \right]$$

с постоянным α фактически веса затухают экспоненциально:

$$Q_k = Q_{k-1} + \alpha \left[r_k - Q_{k-1} \right] = \alpha r_k + (1-\alpha) Q_{k-1} =$$

$$= \alpha r_k + (1-\alpha)\alpha r_{k-1} + (1-\alpha)^2 Q_{k-2} = (1-\alpha)^k Q_0 + \sum_{i=1}^{k} \alpha (1-\alpha)^{k-i} r_i.$$

Нестационарная задача

- Такое правило апдейта не обязательно сходится, но это и хорошо – мы хотим следовать за целью.
- Общий результат – правило апдейта сходится, если последовательность весов удовлетворяет

$$\sum_{k=1}^{\infty} \alpha_k(a) = \infty \quad \text{и} \quad \sum_{k=1}^{\infty} \alpha_k^2(a) < \infty.$$

- Например, для $\alpha_k(a) = \frac{1}{k_a}$ явно сходится.
- Теоретический подход к бандитам:
  - динамическим программированием можно оценить $V^*(n_1, w_1, \dots, n_k, w_k)$ от конца к началу;
  - обобщение — индексы Гиттинса; вычислительно очень сложно.

Стратегии, минимизирующие regret

Динамическое программирование

- Предположим, что агент действует на протяжении h шагов.
- Используем байесовский подход для определения оптимальной стратегии.
- Начинаем со случайных параметров $\{p_i\}$, например, равномерно распределённых, и вычисляем отображение из *belief states* (состояния после нескольких раундов обучения) в действия.
- Состояние выражается как $\mathcal{S} = \{n_1, w_1, \dots, n_k, w_k\}$, где каждого бандита i запустили n_i раз и получили w_i единичек (считаем, что результат бинарный).

Динамическое программирование

- $V^*(\mathcal{S})$ — ожидаемый оставшийся выигрыш.
- Рекурсивно: если $\sum_{i=1}^{k} n_i = h$, то больше нечего делать, и $V^*(\mathcal{S}) = 0$.
- Если знаем V^* для всех состояний, когда осталось t запусков, сможем пересчитать и для $t+1$:

$$V^*(n_1, w_1, \dots, n_k, w_k) = \\ = \max_i \left(\rho_i(1 + V^*(\dots, n_i + 1, w_i + 1, \dots)) + \right. \\ \left. (1 - \rho_i)V^*(\dots, n_i + 1, w_i, \dots)\right),$$

где ρ_i — апостериорные вероятности того, что действие i оправдается (если изначально p_i равномерно распределены, то $\rho_i = \frac{w_i+1}{n_i+2}$).

Байесовский подход к многоруким бандитам

- А теперь давайте посмотрим на многоруких бандитов в общем вероятностном виде.
- Для простоты – бинарный случай, выплата либо 1, либо 0.

Байесовский подход к многоруким бандитам

- Пусть во время t у нас состояние $s_t = (s_{1t}, \dots, s_{Kt})$ для K ручек, и мы хотим дёрнуть такую ручку, чтобы максимизировать общее ожидаемое число успехов.
- Есть функция вознаграждения $R_i(s_t, s_{t+1})$ – награда за дёргание ручки i (a_i), которое переводит состояние s_t в s_{t+1}.
- Есть вероятность перехода $p\left(s_{t+1} \mid s_t, a_i\right)$.
- И мы хотим обучить стратегию $\pi(s_t)$, которая возвращает, какую ручку дёргать.

Байесовский подход к многоруким бандитам

- Тогда value function в самом общем виде до горизонта T:

$$V_T(\pi, s_0) = \mathrm{E}\left[R_{\pi(s_0)}(s_0, s_1) + V_{T-1}(\pi, s_1)\right] =$$
$$= \int p\left(s_1 \mid s_0, \pi(s_0)\right)\left[R_{\pi(s_0)}(s_0, s_1) + V_{T-1}(\pi, s_1)\right] ds_1.$$

- Если всё известно, и T невелико, то можно, опять же, динамическим программированием.
- Но даже подсчитать отдачу от фиксированной стратегии может быть очень дорого, не говоря уж об оптимизации.

Байесовский подход к многоруким бандитам

- Если T большой/неограниченный, логично рассмотреть

$$R = R(0) + \gamma R(1) + \gamma^2 R(2) + \dots, \quad 0 < \gamma < 1.$$

- Теорема Гиттинса (1979): задачу поиска оптимальной стратегии

$$\pi(s_t) = \arg\max_\pi V(\pi, s_t = (s_{1t}, \dots, s_{Kt}))$$

можно факторизовать и свести к

$$\pi(s_t) = \arg\max_i \gamma(s_{it}).$$

- $\gamma(s_{it})$ – *индекс Гиттинса*; но его подсчитать тоже обычно трудно; есть приближения.

Оценки на regret

- Другой вариант – давайте рассчитаем приоритет каждой ручке i так, чтобы непосредственно regret ограничить.
- [Auer et al., 2002]: стратегия UCB1. Учитывает неопределённость, «оставшуюся» в той или иной ручке, старается ограничить regret. Если мы из n экспериментов n_i раз дёрнули за i-ю ручку и получили среднюю награду $\hat{\mu}_i$, алгоритм UCB1 присваивает ей приоритет

$$\mathrm{Priority}_i = \hat{\mu}_i + \sqrt{\frac{2\log n}{n_i}}.$$

Дёргать дальше надо за ручку с наивысшим приоритетом.

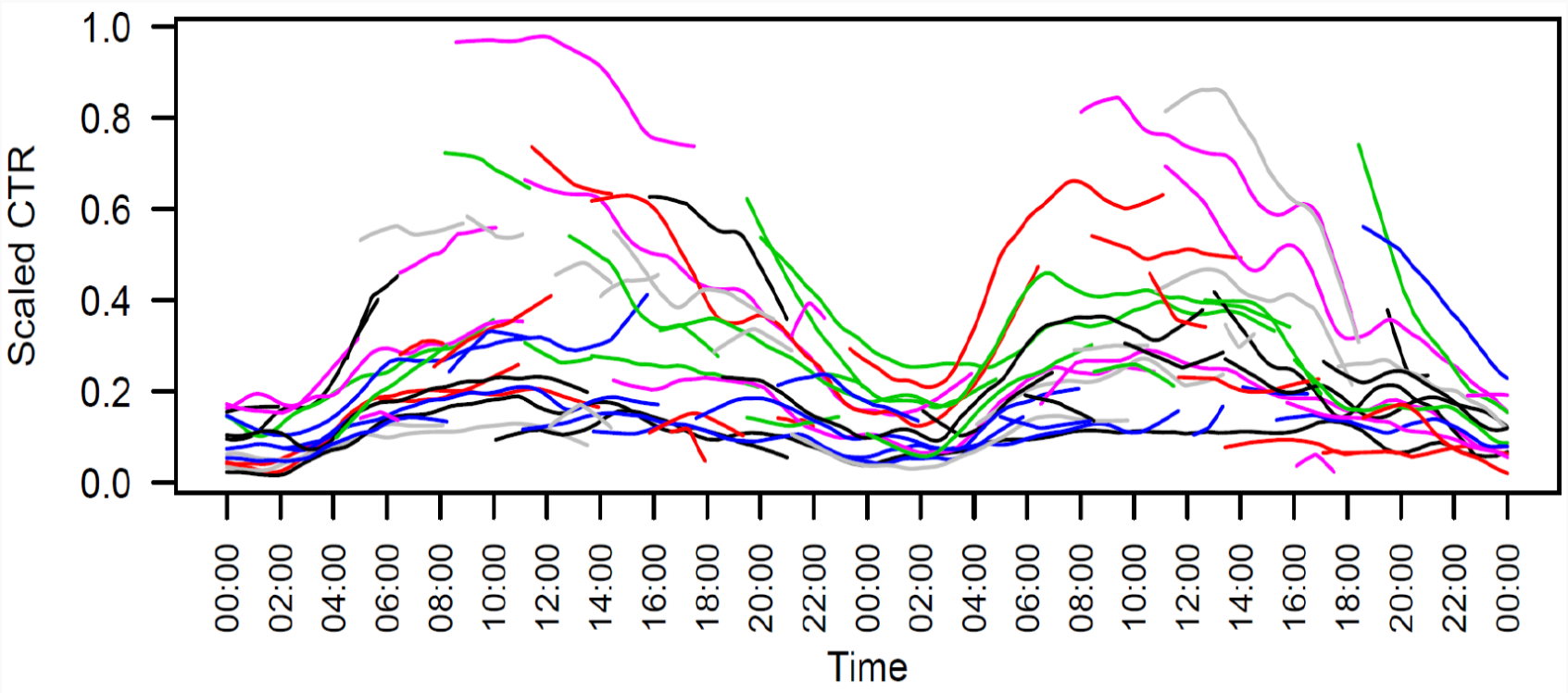
Оценки на regret

- При таком подходе субоптимальные ручки будут дёргать $O(\log n)$ раз, и regret будет $O(\log n)$, а меньше и нельзя (но тут константы тоже важны :)).
- UCB1 – хорошая стратегия, но рассчитывает на оценку в худшем случае, может быть неоптимально.

Клики на странице новостей

Пример: клики на странице новостей

- Пример:



Пример: клики на странице новостей

- Мы в момент t должны распределить доли показов $(x_1, x_2, \dots, x_K)$ так, чтобы оптимизировать CTR.
- Совсем простая ситуация: два момента времени, $t = 0$ и $t = 1$, выбор из двух объектов:
  - объект P имеет CTR p_0 в момент $t = 0$ и p_1 в момент $t = 1$, мы на этот счёт не уверены, есть распределение какое-то;
  - про объект Q всё знаем точно, q_0 и q_1.
- Надо найти x – долю показов P в момент $t = 0$; у нас есть N_0 показов для распределения в $t = 0$ и N_1 в $t = 1$.

Пример: клики на странице новостей

- Пусть мы получили c кликов после того как выбрали x; c – случайная величина.
- Мы пронаблюдаем c, и на втором этапе оптимальное решение будет понятно: дать все N_1 кликов P iff

$$\hat{p}_1(x, c) = \mathrm{E}\left[p_1 \mid x, c\right] > q_1.$$

- Т.е. мы должны оптимизировать x с точки зрения общего ожидаемого числа кликов, учитывая, что что-то новое мы узнаем про p_1 к моменту $t = 1$.

Пример: клики на странице новостей

- Ожидаемое число кликов:

$$N_0 x\hat{p}_0 + N_0(1-x)q_0 + N_1 \mathrm{E}_c\left[\max\{\hat{p}_1(x,c), q_1\}\right] =$$
$$= N_0 q_0 + N_1 q_1 + N_0 x(\hat{p}_0 - q_0) + N_1 \mathrm{E}_c\left[\max\{\hat{p}_1(x,c) - q_1, 0\}\right].$$

- Второе слагаемое – это и есть выгода от исследования P:

$$\mathrm{Gain}(x, q_0, q_1) = N_0 x(\hat{p}_0 - q_0) + N_1 \mathrm{E}_c\left[\max\{\hat{p}_1(x,c) - q_1, 0\}\right],$$

его мы и оптимизируем по x.

Пример: клики на странице новостей

- Если приблизить $\hat{p}_1(x, c)$ нормальным распределением (разумно по ЦПТ):

$$\begin{aligned}
\mathrm{Gain}(x, q_0, q_1) = N_0 x(\hat{p}_0 - q_0) + N_1 \Bigg[\sigma_1(x)\Phi\left(\frac{q_1 - \hat{p}_1}{\sigma_1(x)}\right) + \\
+ \left(1 - \Phi\left(\frac{q_1 - \hat{p}_1}{\sigma_1(x)}\right)\right)(\hat{p}_1 - q_1)\Bigg],
\end{aligned}$$

$$p_1 \sim \mathrm{Beta}(a, b) \text{ (априорное распределение)},$$

$$\hat{p}_1 = \mathrm{E}_c\left[\hat{p}_1(x, c)\right] = \frac{a}{a+b},$$

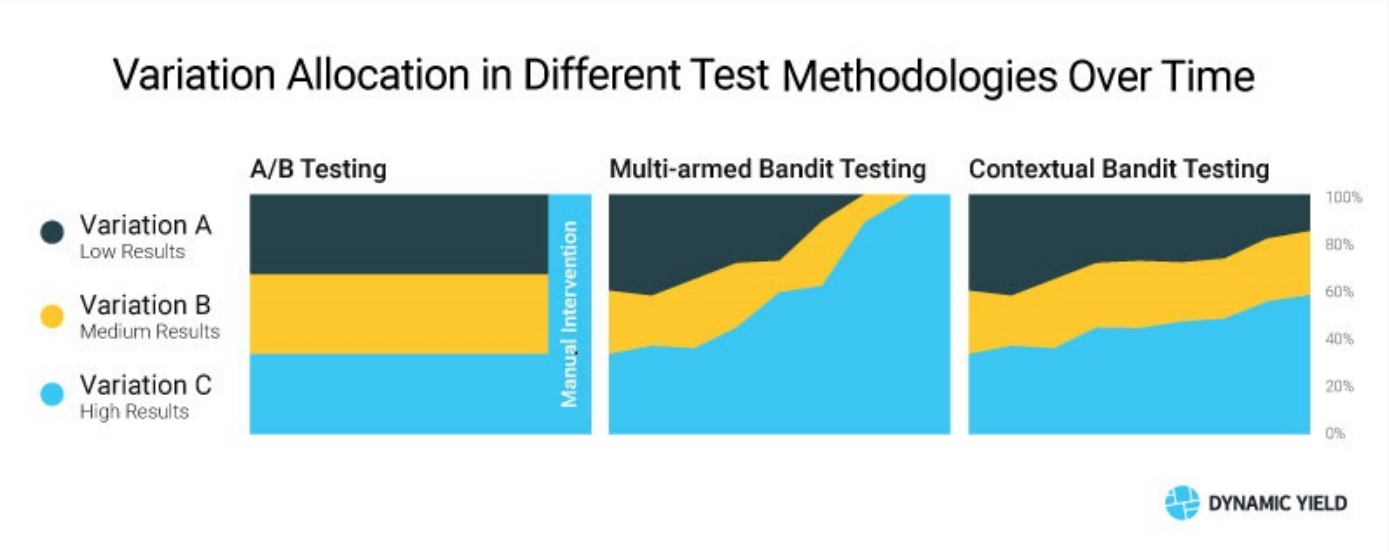
$$\sigma_1^2(x) = \mathrm{Var}\left[\hat{p}_1(x, c)\right] = \frac{xN_0}{a+b+xN_0}\frac{ab}{(a+b)^2(1+a+b)}.$$

Пример: клики на странице новостей

- Для нескольких вариантов ($K > 2$) задача становится гораздо сложнее; её можно несколько ослабить и свести к K независимым задачам с двумя вариантами.
- А что для нескольких временных слотов ($T > 1$)?

Зачем нужны многорукие бандиты

- Многорукие бандиты используются, например, для A/B тестирования.



- Можно и для оптимизации гиперпараметров в тех же нейронных сетях (или где угодно).
- Но для нас сейчас это первый шаг, упрощённая постановка...

Спасибо!

Спасибо за внимание!